UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería de Tecnologías y Servicios de la Telecomunicación

TRABAJO FIN DE GRADO

Detección jerárquica de grupos de personas con CNNs

Antonio Campoy Cordero
Tutor: Álvaro García Martín
Ponente: José María Martínez Sánchez

Junio 2019

Detección jerárquica de grupos de personas con CNNs

AUTOR: Antonio Campoy Cordero
TUTOR: Álvaro García Martín

Video Processing and Understanding Lab, VPU
Dpto. 111
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio 2019

Trabajo parcialmente financiado por el Gobierno de España bajo el proyecto TEC2017-88169-R (MobiNetVideo)

Resumen (castellano)

En la actualidad, no es raro sentirnos observados por cámaras mientras hacemos vida normal por las calles o en lugares cerrados. Y es que una cámara grabando 24 horas al día, 365 días al año, genera una información de la cual podemos, si sabemos tratarla, obtener una inmensa cantidad de datos valiosos.

A día de hoy, ya hay personas que se han preocupado de tratar esos datos, generando algoritmos que permiten detectar a las personas que pasan por delante de las cámaras, y aunque estos ya son bastantes sofisticados, todos se topan con el mismo problema, la oclusión de unas personas con otras en las imágenes capturadas. Por ello, se han desarrollado algoritmos que mejoran la captura de estas personas, y aunque los resultados son satisfactorios, no lo son lo suficiente.

Además en los últimos años, la utilización de Inteligencia Artificial, en concreto, las Redes Neuronales Convolucionales (CNNs) para el tratamiento de imágenes, ha conseguido mejorar estos algoritmos, que por lo general usaban técnicas tradicionales como el descenso por gradiente. Los resultados son muy alentadores en todos los ámbitos de detección de objetos, personas y otros elementos, en imágenes.

Es por ello, que proponemos en este Trabajo de Fin de Grado mejorar un algoritmo basado en Deformable Parts Model (DPM) y CNNs, con el objetivo de añadir esos casos en los que nos encontramos con oclusión de personas por otras, usando un modelo que no solo se centra en la persona principal sino que analiza su entorno más próximo en busca de más personas que puedan estar ocluidas y en las que solo vemos, por ejemplo, la mitad derecha del cuerpo.

El objetivo principal será mejorar el algoritmo ya existente, evaluado sobre ciertas secuencias de vídeo representativas siendo el resultado satisfactorio.

Palabras clave (castellano)

CNN, DPM, Convolucionales, detección de personas, oclusiones, jerarquía, HDG, DTDP, DP-DPM

Abstract

At present, it is not uncommon to feel watched by cameras while we are making normal life by the streets or in closed places. A camera that records 24 hours a day, 365 days a year, generates information of which, if we know how to treat it, we obtain a large amount of valuable data.

Nowadays, there are some people who have worried about the treatment of this data, generating algorithms that allow detect the people who pass in front of the cameras, although that algorithms are already sophisticated, all of them have the same problem, the occlusion between persons at the captured images. Thus, they have been developed algorithms that improves the capture of this people, even though the results are satisfactory, there aren´t enough.

Also in the last years, the use of the Artificial Intelligence, in concrete, Convolutional Neuronal Networks (CNNs) for the treatment of images, has managed to improve these algorithms, which usually used traditional techniques such as gradient descent. The results are very encouraging in all areas of detection such an objects, people, and another elements in images.

This is why we propose in this Bachelor's Degree Final Work to improve an algorithm based on Deformable Parts Model (DPM) and CNNs, with the aim of adding those cases in which we find occlusion of people by others, using a model that not only focuses on the main person but also analyzes its closest environment in search of more persons that may be occluded and in which we only see, for example, the right half of the body.

The main objective will be to improve the existing algorithm, evaluated through certain representative video sequences being the result satisfactory.

Keywords (inglés)

CNN, DPM, Convolutional, detection of persons, occlusions, hierarchical, HDG, DTDP, DP-DPM

Agradecimientos

Si echamos la mirada atrás, y vemos al Antonio que por primera vez pisaba esta maravillosa facultad, nos damos cuenta de inmediato que algo ha sucedido con él. Uno no sabe que es lo que le deparará el futuro, y es por ello que cuando se tiene esa incertidumbre lo mejor es rodearse de esas personas que sabes que siempre van a estar a tu lado apoyándote.

En primer lugar quería agradecer a mi familia los valores que me han infundado desde el día que nací, a mi madre por saber trasmitirme ese coraje para enfrentarme a los problemas, a mi padre por enseñarme que en esta vida hay que tener mano izquierda para todo, y por último, a mi hermana, a la que siempre está ahí y que es capaz de sacarme una sonrisa cuando las cosas no van bien. Gracias.

En segundo lugar, y no menos importante, a las personas con las que he compartido mis logros dentro de la carrera y los desayUAM de todas las mañanas. Sergio Vivas, es la persona a la que siempre puedes tomar como ejemplo a pesar de su carácter, Ricardo Hawtin SUPREME, como bien dijo, compañero de derrapes tanto dentro como fuera de la carretera, y David Abreu la persona que siempre sabes que va a estar cuando le necesites con una sonrisa en la cara. También está esa persona con la que probablemente he pasado más tiempo que con nadie a lo largo de mi vida, Javier Galán, no hace falta hablarnos para saber lo que pensamos. Y Diego Herrero mi ángel de la guarda. Gracias a estas personas, hoy en día tengo el honor de formar parte de un gran proyecto como Génova.

Hay personas en la vida que aparecen el día menos esperado y de repente pasan a ser un pilar fundamental de tu día a día. Paula Marino, gracias por enseñarme valores que desconocía hasta el día de hoy.

Por último, agradecer a Álvaro García Martín, probablemente uno de los profesores que mejor me han sabido trasmitir sus amplios conocimientos de Teleco y que además he tenido la suerte de que sea mi tutor del TFG. Gracias por siempre contestar mis infinitas dudas. También agradecer a José María Martínez por ser mi ponente.

¡Muchas gracias!

INDICE DE CONTENIDOS

1 Introducción 1
1.1 Motivación 1
1.2 Objetivo 2
1.3 Organización de la memoria 2
2 Estado del arte 3
2.1 Introducción 3
2.2 Discriminatively Trained Deformable Part-based model (DTDP) 3
2.3 Hierarchical detection of persons in groups (HDG) 7
2.4 Deep Pyramid – Deformable Parts Model (DP-DPM) 10
2.5 Conclusiones 12
3 Diseño 15
3.1 Introducción 15
3.1.1 Lenguaje de programación 15
3.1.2 Modelo 15
3.1.3 Modelo jerárquico de grupos 17
3.1.4 Modelo jerárquico de partes 17
3.1.5 Distribuciones 18
3.1.6 Almacenamiento de resultados 19
3.2 Conclusiones 19
4 Desarrollo 21
4.1 Introducción 21
4.2 Implementación 21
4.2.1 Instalación de software 21
4.2.2 Jerarquía de grupos 22
4.2.3 Jerarquía de partes 22
4.2.4 Distribuciones 24
4.2.5 Eliminación de redundancia 26
4.3 Conclusiones 28
5 Integración, pruebas y resultados 29
5.1 Introducción 29
5.2 Elección de dataset 29
5.3 Evaluación 29
5.3.1 Hierarchy of persons in groups (HPG) 30
5.3.2 Hierarchy of body parts (HBP) 31
5.3.3 Hierarchical detection in groups (HDG) 32
5.4 Comparación de algoritmos 34
5.5 Conclusiones 34
6 Conclusiones y trabajo futuro 35
6.1 Conclusiones 35
6.2 Trabajo futuro 35
Referencias 37
Glosario 38

INDICE DE FIGURAS

FIGURA 2-1 IMAGEN DEL MODELO QUE USA DTDP Y EJEMPLO DE DETECCIÓN. [1] 4

FIGURA 2-2 PIRÁMIDE QUE MUESTRA EL MODELO DE PERSONA. CUANDO LA PIRÁMIDE ALCANZA LA DOBLE RESOLUCIÓN SE APLICAN LOS FILTROS DE PARTES. [1] 5

FIGURA 2-3 MAPA DE LA EXTRACCIÓN DE LA IMAGEN DE DETECCIONES. DESDE EL INPUT AL OUTPUT. [1].....6

FIGURA 2-4 NUEVO MODELO USADO EN HDG PARA PERSONA. [8] 7

FIGURA 2-5 IMAGEN DE LAS ZONAS DE BÚSQUEDA DEL MODELO QUE PROPONE HDG Y SU JERARQUÍA DE PERSONAS. [8] 8

FIGURA 2-6 DIFERENTES CONFIGURACIONES DEL MODELO DE PERSONA QUE UTILIZA HDG.[2] 9

FIGURA 2-7 CONFIGURACIÓN DEL MODELO DE PERSONA QUE UTILIZA PARA DETECTAR A SP.[8] 9

FIGURA 2-8 ESQUEMA DEL FUNCIONAMIENTO DE LA DEEP PYRAMID DPM.[3].....10

FIGURA 2-9 ESQUEMA DE FUNCIONAMIENTO DE LA DPM-CNN. [3].....11

FIGURA 3-1 MODELO DE PERSONA UTILIZADO POR DP-DPM Y QUE UTILIZAREMOS EN DP-HDG. 16

FIGURA 3-2 CONFIGURACIONES ELEGIDAS PARA EL NUEVO MODELO. 18

FIGURA 4-1 FLUJO QUE SEGUIRÁ UNA IMAGEN DESDE QUE ENTRA HASTA SU SALIDA. 21

FIGURA 4-2 FUNCIÓN DE DISTRIBUCIÓN DE SUMA ACUMULADA DEL CUERPO ENTERO. 25

FIGURA 4-3 SOLAPAMIENTO ENTRE DOS CABEZAS.....27

FIGURA 4-4 SOLAPAMIENTO ENTRE DOS CAJAS DE CUERPO ENTERO 27

FIGURA 5-1 CURVA ROC DE LA SECUENCIA S2L1 CON HPG.....30

FIGURA 5-2 CURVA ROC PARA LA SECUENCIA S2L1 CON HBP.....32

FIGURA 5-3 CURVA ROC PARA LA SECUENCIA S2L1 CON HDG.....33

INDICE DE TABLAS

TABLA 1: RESULTADOS HPG PARA LA SECUENCIA S2L1.....30

TABLA 2 RESULTADOS CON LAS DIFERENTES CONFIGURACIONES PARA LOS DIFERENTES VIDEOS. 31

TABLA 3 RESULTADOS HBP PARA LAS DIFERENTES SECUENCIAS.....31

TABLA 4 RESULTADOS DE HDG PARA LAS DIFERENTES SECUENCIAS.....33

TABLA 5 RESULTADOS DE TODOS LOS ALGORITMOS.....34

1 Introducción

1.1 Motivación

En pleno siglo XXI, podemos afirmar que es la era más tecnológica de todos los tiempos, y es por ello que la creciente demanda de grabaciones en las ciudades ha aumentado considerablemente la información en forma de imágenes que tenemos disponible.

Toda esta información que se genera cada segundo, si se procesa adecuadamente, puede ser útil para mejorar, en la medida de lo posible, la sociedad. Imaginemos una cámara que graba veinticuatro horas al día en una calle céntrica de una ciudad concurrida, todo lo que esta cámara captura puede ser de utilidad, por ejemplo, para controlar que no haya una aglomeración de personas y puedan surgir acontecimientos no deseados, o para detectar cuantas personas al día atraviesan esta calle, etc… Son miles las ideas que se nos ocurren.

Existen personas y grupos de investigación que se han preocupado por tratar estas imágenes, y que han hecho algoritmos encargados de detectar objetos que se mueven en ellas, y aunque dan buenos resultados, muchos se encuentran con un mismo problema en común, y es cuando estos objetos se solapan unos con otros, es en este caso en el que se produce peor rendimiento, ya que la complejidad de las detecciones de objetos es elevada.

Si nos centramos en la detección de personas, es normal que estas anden por las calles acompañadas, y por lo general, a ojos de una cámara que captura la imagen tendrá una mayor probabilidad de que estás en algún momento se solapen y escondan partes de su cuerpo detrás de otra persona.

Por otro lado, existe una tendencia de uso de Inteligencia Artificial, para mejorar el rendimiento de los algoritmos de detección de objetos, en concreto las CNNs (Convolutional Neuronal Networks), las cuales trabajan eficientemente con imágenes.

Si juntamos la detección de personas, con el uso de CNNs y además le añadimos el reto de mejorar el problema que surge de la oclusión de personas, nos encontramos ante una maravillosa oportunidad de investigación dentro de este campo y por lo tanto, una motivación más que suficiente para realizar este trabajo de fin grado.

1.2 Objetivo

Ahora que la motivación para realizar el trabajo de fin grado es clara, podemos plantear los objetivos para su realización de forma ordenada y progresiva:

1. Estudio del Arte:
   Se procederá a entender el funcionamiento de los diferentes algoritmos de detección de personas que utilizaremos más tarde para el desarrollo de este trabajo. En concreto son tres, en primer lugar DTDP (Discriminatively Trained Part Based Models) [1] el algoritmo del que parte todo, en segundo lugar HDG (Hierarchical Detection of Persons in Groups) [2] y por último DP-DPM (Deep Pyramid Deformable Part Models) [3] .

2. Instalación del software:
   Se instalará la herramienta que se utiliza para ejecutar los diferentes algoritmos de partida e implementar el nuestro. En el caso del DTDP y HDG, el software se trata de Matlab R2016b, para el algoritmo DP-DPM y el nuestro se utiliza Caffe [6], un framework de aprendizaje profundo a través de Matlab.

3. Realización del nuevo algoritmo:
   En base a lo aprendido de los algoritmos anteriores y a los requisitos necesarios para resolver el problema de la oclusión de personas, se desarrolla el nuevo algoritmo.

4. Obtención de resultados:
   Se prueba el algoritmo generado con una base de datos de videos, de diferente complejidad, ambiente y tamaño de las personas.

5. Realización de la memoria:
   Por último se procede a realizar la memoria del trabajo de fin grado en el cual queda reflejado todo el proceso de elaboración e información referente al funcionamiento y logística del algoritmo.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

1. Introducción
2. Estado del arte
3. Diseño
4. Desarrollo
5. Integración, pruebas y resultados.
6. Conclusiones y trabajo futuro.

2 Estado del arte

2.1 Introducción

Como ya hemos dicho, la detección de personas ya se ha tratado con anterioridad, en este apartado de estado del arte hablaremos en profundidad sobre lo que ya hay desarrollado, con la finalidad de ponernos en contexto para más tarde poder entender mejor como se ha desarrollado este nuevo algoritmo. También nos ayudará a saber qué resultados ofrecen y cómo se comportan en diferentes situaciones de detección.

El objetivo principal es implementar un algoritmo de detección de personas basado en CNNs, por lo que describiremos tres algoritmos ya existentes, con gran relación.

Dividiremos esta sección en tres partes, la primera para el algoritmo DTDP [1] del que nace el siguiente, HDG [2] y que se encarga de intentar solucionar el problema de oclusión de personas, estos dos algoritmos utilizan características tradicionales como el HOG [4] (Histogram of Oriented Objects). Por último, hablaremos sobre el algoritmo base DP-DPM [3] del que partimos para la realización del nuestro.

2.2 Discriminatively Trained Deformable Part-based model (DTDP)

Discriminatively Trained Deformable Part-based model es un sistema de detección de objetos basado en mezclas de modelos multiescala deformables, este tiene un sistema de raíz o root, donde este root y cada parte del cuerpo deformable son modeladas mediante HOG. Este sistema es capaz de detectar gran variedad de objetos altamente variables y conseguir buenos resultados.

Como nuestro algoritmo se basa en la detección de personas, hablaremos de este haciendo referencia a su modelo de detección de personas.

Lo que propone, en primer lugar es dividir a una persona en diferentes partes significativas, en este caso la divide en cinco, una parte para la cabeza, otra para tronco y brazo izquierdo, otra para tronco y brazo derecho, otra para pantorrilla y la última para pies. De esta manera se puede centrar en zonas más concretas de una persona, para por último, de la forma que hemos explicado poder detectarlas

En la siguiente Figura 2-1 podemos ver un ejemplo representativo de cómo divide el modelo, en (a) tenemos el filtro del root del modelo, en (b) tenemos un híbrido entre los filtros a la doble resolución que el root y las partes, y en (c) el modelo espacial para cada ubicación de las partes.

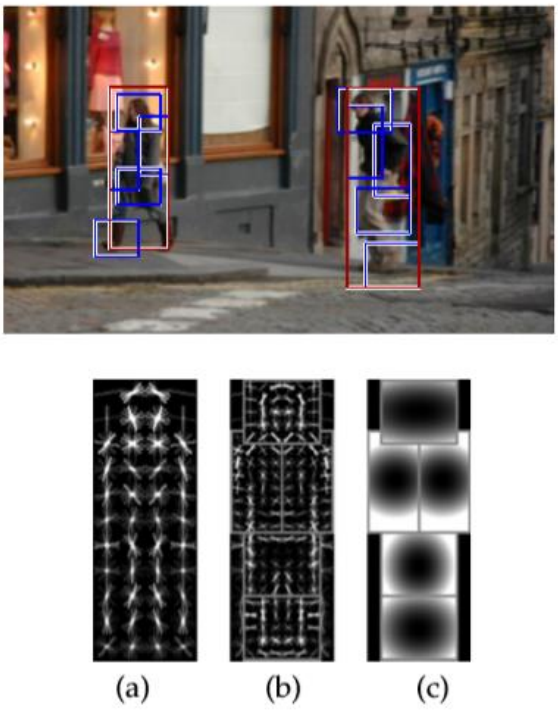

Figura 2-1 Imagen del modelo que usa DTDP y ejemplo de detección. [1]

Antes de entrar más en profundidad con el algoritmo explicaremos que es Histogram of Oriented Gradient.

Histogram of Oriented Gradient [4] es el método encargado de evaluar histogramas sobre una imagen, cada imagen se divide en un número de regiones equivalentes a los que denominamos cell, dentro de cada cell se calcula la orientación del gradiente y la suma de todos los cell nos da como resultado, cómo podemos ver en la Figura 2-1, el (a), con el HOG del filtro del root del modelo.

Con el fin de obtener mejores resultados frente a sombras o diferentes iluminaciones, este algoritmo acumula la suma de la energía de los cells de una determinada zona en blocks y de esta forma proceder a normalizar todos los cells que contiene cada block, son a estos blocks a los que llaman descriptores HOG, los cuales pueden solaparse entre ellos y que más tarde mediante SVM (Support Vector Machine) [1] se clasificarán como una detección positiva o negativa. Como resultado de este método obtenemos una representación acertada de los contornos de un objeto que además es fiel a variaciones leves de los objetos, en este caso las personas, obteniendo buenos resultados a costa de un gran coste computacional.

Una vez explicado HOG podemos seguir con DTDP, y una vez definimos las partes del modelo (n=0,…, N) diferenciamos 3 variables, las cuales llamamos F_n como la respuesta del filtro HOG para cada parte, por otro lado $v_{n,0}$ será el vector que define la posición de esa parte respecto del root y que es de dos dimensiones, y por último d_n que especifica los coeficientes de la función cuadrática de la deformación que pueda tener cada parte y es de cuatro dimensiones. También entra en juego s que indica la escala. Por lo que sabiendo lo que significa cada una de las variables anteriormente dichas, podemos pasar a explicar cómo se consigue la puntuación $BP_n(x,y,s)$ que podemos definir como la puntuación de un pixel para una parte determinada a una escala concreta.

Para ello usa la siguiente fórmula:

$$BP_n(x, y, s) = F_n(x, y, s) - \langle d_n, \phi(dx_n, dy_n) \rangle$$

En primer lugar, calcula el desplazamiento de la parte n en realición con el ancho:

$$(dx_n, dy_n) = (x_n, y_n) - (2(x_0, y_0) + v_{n,0})$$

Ahora calcula las posibles distribuciones de la deformación espacial que pueda sufrir cada parte:

$$\phi(dx, dy) = (dx, dy, dx^2, dy^2)$$

Una vez se obtiene el BP_n, se procede a hacer el sumatorio del root y las N partes del cuerpo por cada pixel y escala que tengamos para obtener C(x,y,s) o puntuación final de la detección, definida por la siguiente ecuación:

$$C(x, y, s) = \sum_{n=0}^{N} BP_n(x, y, s)$$

Este modelo usa además lo que llaman pirámide de carácterísticas, como podemos ver en la Figura 2-2, son diferentes niveles de la imagen redimensionada hasta llegar a cierta resolución, en cada una se calculan los scores y por último se suman para dar como resultado el score final.

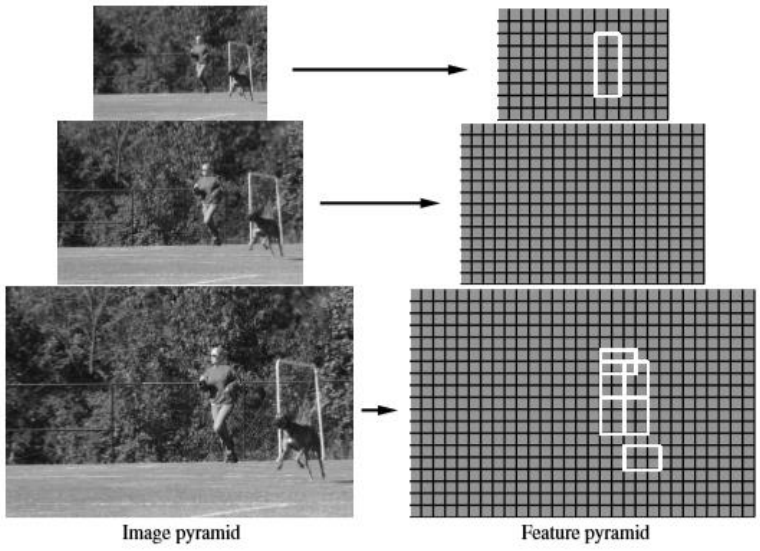

Figura 2-2 Pirámide que muestra el modelo de persona. Cuando la pirámide alcanza la doble resolución se aplican los filtros de partes. [1]

Una vez obtenida la detección, se define un umbral, se detecta sobre este umbral y se pasa por otro llamado NMS (Non-Maximum Suppression) que se encarga de descartar aquellas detecciones que podemos considerar como repetidas, el valor del umbral define en cuanto tanto por ciento de solape entre dos detecciones consideramos que es repetida. Es decir, si ponemos un umbral 0.5 de NMS estaremos diciendo que si estas dos detecciones se

solapan en un 50% o más descartaremos la de peor puntuación, ya que será una detección redundante.

A modo recapitulatorio, en la Figura 2-3 podemos observar de forma más visual y ordenado todo lo que acabamos de ver, entra una imagen al algoritmo que se divide en dos caminos, uno para el mapa de características y otro para el mapa de características a doble resolución. Al mapa de características se le multiplica por el modelo de persona root la cual dará como resultado unos scores que podemos ver respresentado en color, siendo colores oscuros scores bajos y colores claros detecciónes altas. Por el otro lado, para cada n (parte del cuerpo) se multplica por el mapa de caracteristicas a doble resolución, que da al igual que en el caso anterior unos scores los cuales podemos ver representados por color.

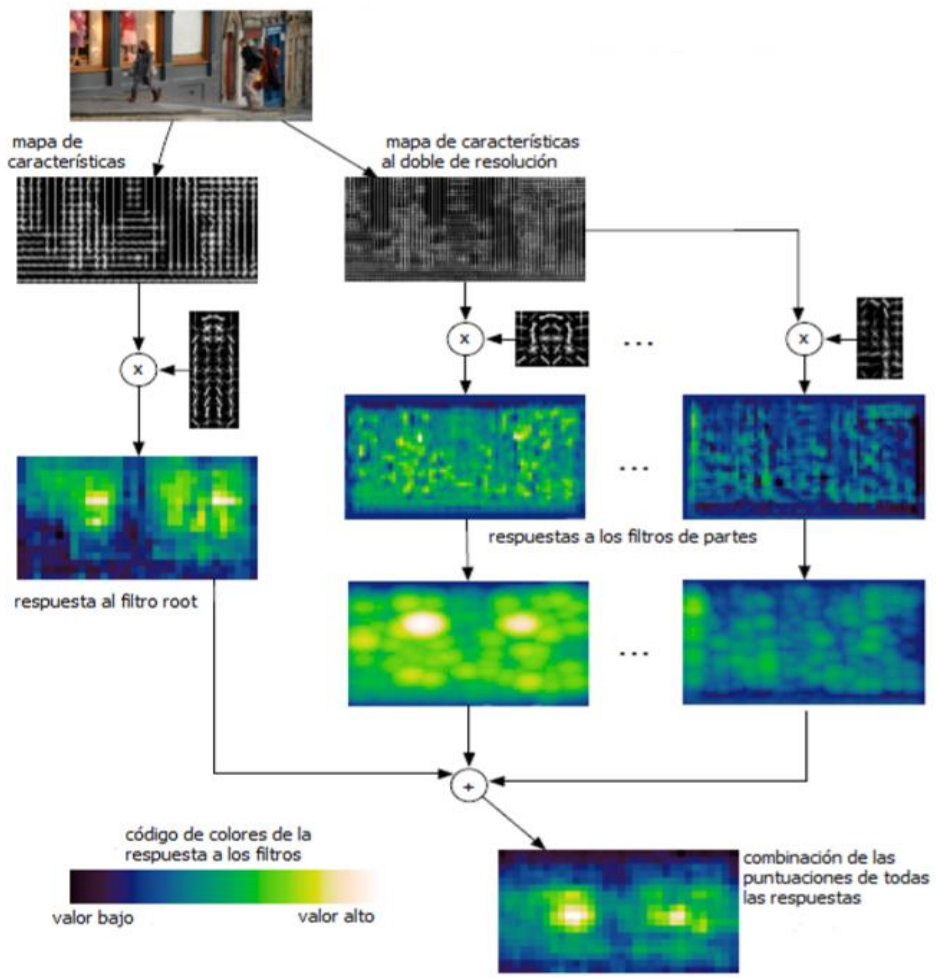

Figura 2-3 Mapa de la extracción de la imagen de detecciones. Desde el input al output. [1]

Este algoritmo DTPD, como hemos dicho, ofrece unos resultados aceptables usando descriptores HOG, sin embargo el problema con el que se encuentra es cuando la persona a detectar se encuentra ocluida, ya sea por otra persona o por un objeto existente en la imagen, ya que esas partes del cuerpo que se encuentran ocluidas no suman al resultado global de todas las partes dando una puntuación muy baja y que probablemente se quede por debajo del umbral pedido para considerar una detección como positiva.

2.3 Hierarchical detection of persons in groups (HDG)

Como ya sabemos, el problema principal que presenta DTDP se produce cuando hay escenarios con múltiples personas las cuales se solapan unas con otras perdiendo información de partes del cuerpo y que por lo tanto reduce el número de personas detectadas. Este algoritmo denominado HDG [2] (Hierarchical detection of person in groups) propone mejorar el algoritmo base DTDP para solucionar el problema del que hablamos, mediante diferentes configuraciones de partes del cuerpo de personas en segundo plano, el cual desarrollaremos a continuación.

HDG propone varias mejoras sobre el algoritmo DTDP, para como decíamos, mejorar el problema de la oclusión. Lo que primero propone es crear una jerarquía de grupos, y de esta forma usar la información de la persona principal de modo que a su alrededor dentro de unos límites poder detectar a personas parcialmente ocluidas, para ello utiliza un modelo de persona diferente, este modelo es el *INRIA person 2007 rc16* [1] , este modelo a diferencia del de DTDP usa a parte del root, persona principal, ocho partes del cuerpo que podemos ver en la siguiente Figura 2-4, y que modelan mejor a una persona y sus partes del cuerpo, las cuales son deformables al igual que en el DTDP. Además este modelo tiene flip, es decir, el modelo de una persona vista de frente y por detrás.

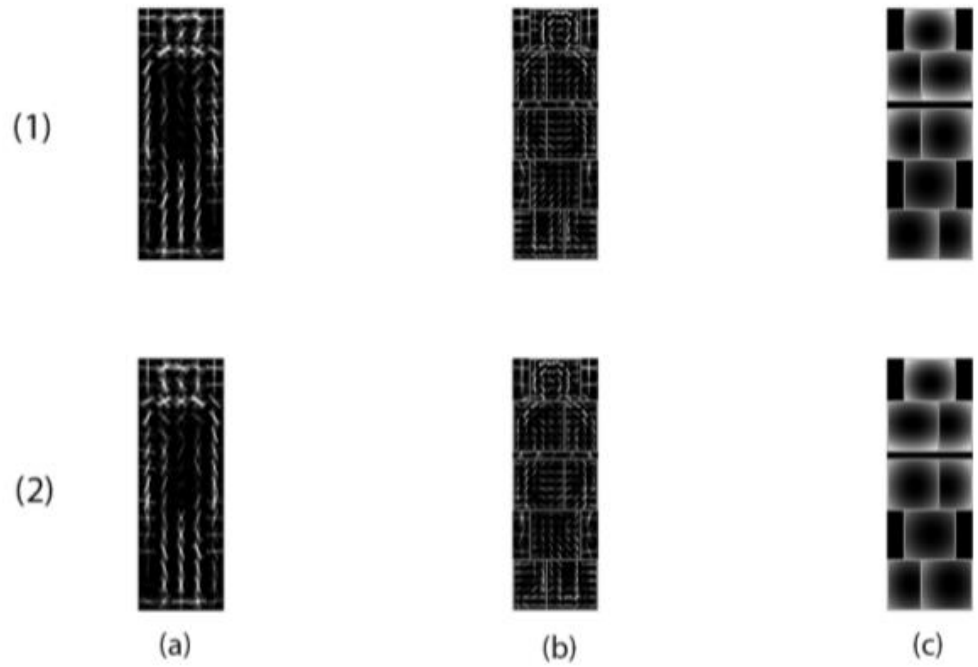

Figura 2-4 Nuevo modelo usado en HDG para persona. [8]

Como se usa jerarquía de personas, diferencia dos tipos, la que llama MP (Main Person) y la SP (Secondary Person). Por lo que propone la variable *anchor_shift* que tiene la información de la distancia (Δx, Δy) a la que están las SP respecto al centro del root de la MP, de esta forma recopila información de las personas que están a su alrededor para mejorar su puntuación de detección en función de la MP.

Con el fin de hacer un barrido alrededor de la MP, *anchor_shift* será una variable con nueve vectores, nacidos de la combinación del valor que se le pone. En este caso el autor propone los valores -6 y +6 para el eje horizontal, ya que es aquella distancia que hace que las oclusiones de las SP, frente a la MP, sean de la mitad de su cuerpo. Y en el caso del eje vertical -6 y +6 ya que es la distancia a la que SP se le ve la cabeza por encima de la MP. En la Figura 2-5 podemos ver de forma representativa las búsquedas que realiza de SP alrededor de la MP

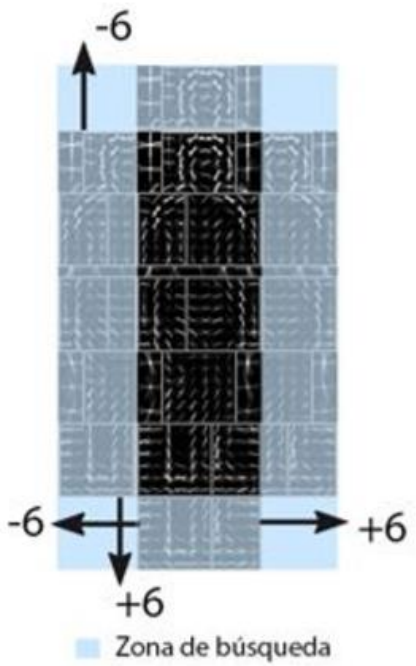

Figura 2-5 Imagen de las zonas de búsqueda del modelo que propone HDG y su jerarquía de personas. [8]

A parte de buscar a la persona completa y con todas sus partes, también puede suceder que la persona principal se encuentre ocluida por algún objeto, por lo que también lo que propone este algoritmo son diferentes configuraciones de la MP y las SP, que sea diferente a la principal, en concreto propone cinco configuraciones que podemos ver en la Figura 2-6, las cuales van quitando partes del cuerpo de abajo a arriba, la configuración 11 corresponde a la del cuerpo completo, es decir, la original; a la configuración 12 se le quitan los dos pies, para la configuración 13 se le quitan los pies y la pantorrilla, en la configuración 14 solo nos queda la cabeza y los hombros, el conjunto de estas tres partes es característica por formar una omega Ω como resultado de aplicarle el gradiente, y por último la configuración 15 que solo tiene la cabeza de la persona. Al igual que antes independiente de la configuración elegida, se buscará alrededor de la MP con su configuración correspondiente.

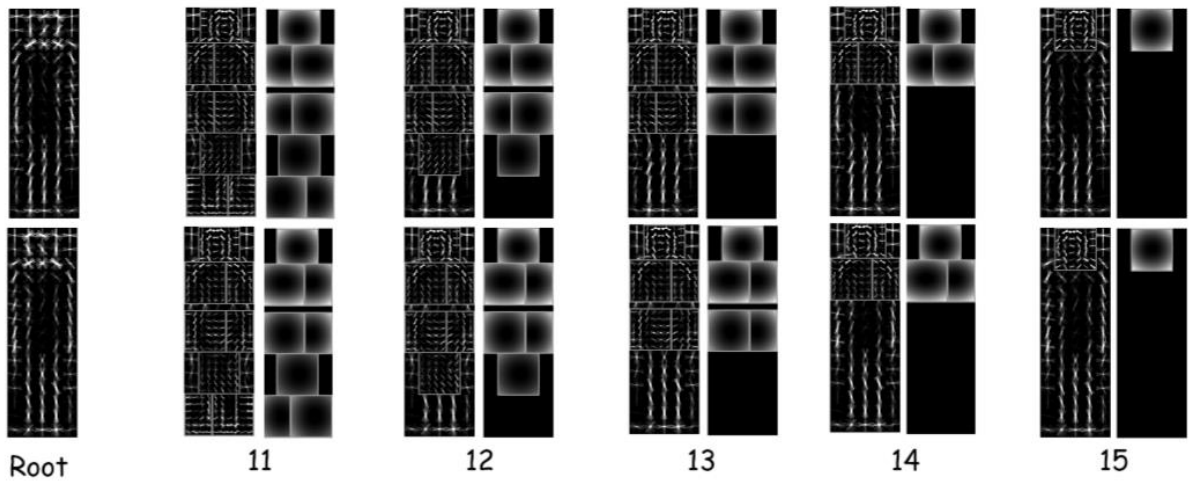

Figura 2-6 Diferentes configuraciones del modelo de persona que utiliza HDG.[2]

Este modelo, para cada configuración, busca de forma diferente alrededor de la persona, es decir, si por ejemplo estamos buscando con la configuración 11 a una persona que se encuentra a la izquierda, la configuración de las partes será diferente, semejante a la forma de la Figura 2-7

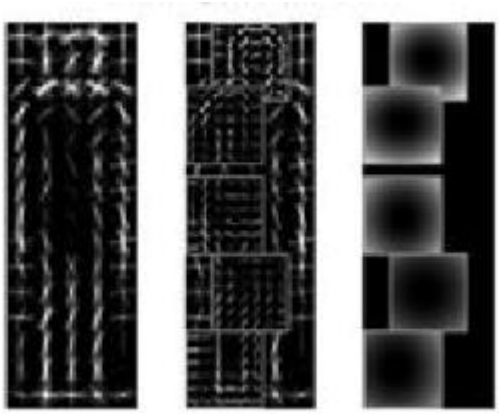

Figura 2-7 Configuración del modelo de persona que utiliza para detectar a SP.[8]

Esto es debido a que si se encuentra por detrás de la MP hay partes que se da por hecho que no se van a localizar y no se molesta en intentar detectarlas, de esta forma, da más peso a las cajas de las partes que sí que están y se obtendrá una detección de esa persona que estaba ocluida y que con el algoritmo original DTDP no se detectaba.

Una vez realizado las detecciones, este algoritmo HDG se encarga al igual que el DTDP de aplicarle un NMS, aunque de forma diferente, en este caso el valor del NMS aplicado es del 90% y por otro lado, se hace otro NMS de la parte de la cabeza con un valor de 0%, lo que quiere decir, es que si se detecta una persona alrededor de la MP y la caja de la detección de las cabezas entre estas dos solapa, aunque sea, mínimamente más de un 0% se

dará por hecho que se trata de la misma persona, quedándose con la cabeza y la caja de detección de la MP.

HDG ofrece unos resultados óptimos, y cumple con sus expectativas, mejorando tanto en precisión (ajuste de las cajas de detección) como en recall (personas detectadas) para secuencias de vídeo complejas, y para secuencias de videos más simples no empeora el rendimiento aunque en algunas secuencias lo llega a mejorar.

2.4 Deep Pyramid – Deformable Parts Model (DP-DPM)

En primer lugar, decir que este algoritmo de detección de objetos es una adaptación por parte del autor del DTDP buscando seguir la tendencia de uso de CNNs. Lo que propone es un nuevo algoritmo que usando un modelo de partes deformables y una red neuronal convolucional consiga mejorar los resultados que ofrecía el algoritmo original. Por lo general, los buenos resultados que se obtenían mediante el uso de características tradicionales como HOG, del cual ya hemos hablado, hacen más grande el reto del autor de mejorarlo. Pasemos a ver cuál es su funcionamiento:

El autor empieza introduciendo su algoritmo como una red convolucional que toma como entrada una pirámide de imágenes y produce como salida una pirámide de puntuaciones de detección.

Con el fin de detectar todo tipo de objetos y de todo tipo de tamaños, el autor propone algo parecido a los otros dos algoritmos que generaban una pirámide a diferentes escalas de imagen, en este algoritmo se utiliza la pirámide introduciéndola en la CNN como una pirámide de imágenes y obteniendo una pirámide de características (DeepPyramid) que vuelve a pasar por lo que llama DPM-CNN (Deformable Parts Model – Convolutional Neuronal Network) para conseguir la pirámide de puntuaciones. Para ello usa una arquitectura estándar de escala única [5] y después vincula los pesos de la red en las escalas.

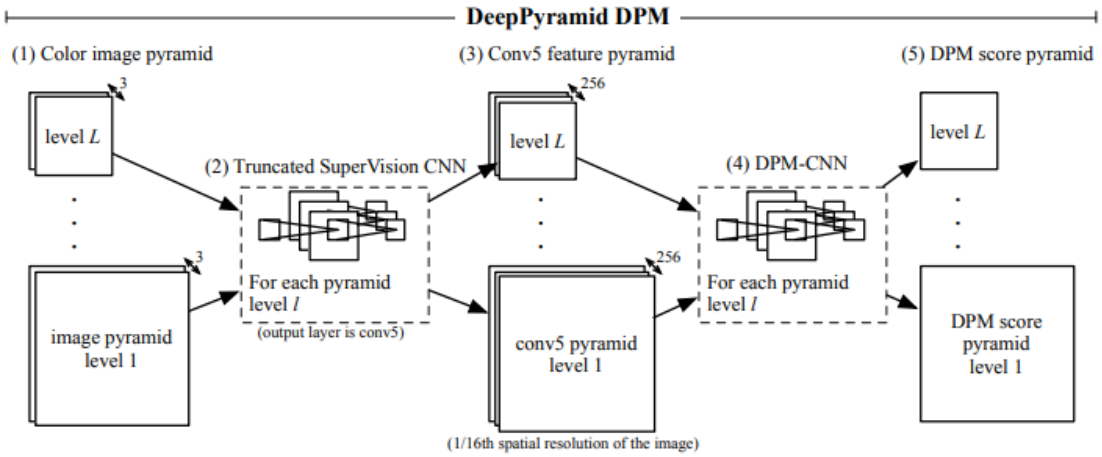

Figura 2-8 Esquema del funcionamiento de la Deep Pyramid DPM.[3]

En la Figura 2-8 podemos ver representativamente como es el proceso de generación de la DeepPyramid y la pirámide de puntuaciones de las detecciones. Como decíamos primero se genera la pirámide de imágenes a partir de la original redimensionándola a diferentes

escalas (1), cada nivel de la pirámide se propaga a través de la red neuronal convolucional (SuperVision CNN) [5] que acaba en la quinta capa convolucional (Conv5) (2), el resultado es una pirámide de características y cada capa está a 1/16 de la resolución de cada nivel de la pirámide (3). Cada nivel de esta última pirámide pasará por lo que el autor llama DPM-CNN (4), y por último lo que obtenemos es la pirámide de puntuaciones de cada nivel de esta. Aunque el sistema parece que tiene dos CNNs, el autor recalca que se puede tratar como una sola.

Este algoritmo también usa modelos de los objetos, cada modelo con su root y sus demás partes que lo conforman, con el fin de detectar lo que deseemos (personas, coches, objetos etc…), una vez obtiene la pirámide de características de la imagen, el funcionamiento de detección es parecido al de los otros dos algoritmos, sin embargo este no usa HOG, por lo que el autor propone convertir el DPM en una CNN equivalente, lo llama DPM-CNN que funciona de la siguiente manera:

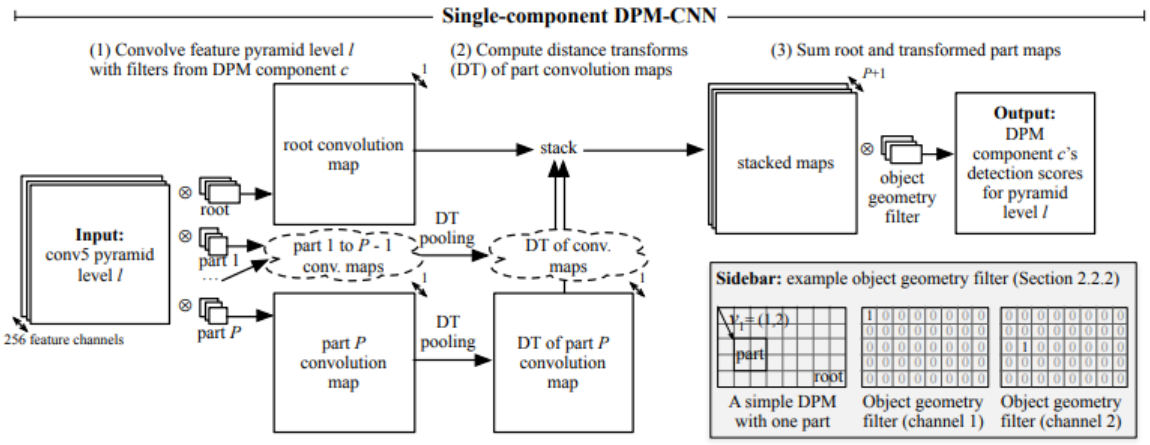

Figura 2-9 Esquema de funcionamiento de la DPM-CNN. [3]

Como podemos ver en la Figura 2-9 en primer lugar entra la pirámide de características calculada anteriormente, y a cada nivel de la pirámide se le pasa por los filtros propuestos por el modelo, el filtro principal root y los filtros de cada una de las partes P, creando un mapa de convolución del root y P mapas de convolución de partes (1), los mapas de las partes son pasados por lo que llama DT (Distance Transform) pooling, que es la agrupación de la transformación de las distancias, que al final lo que hace es una generalización de máximas agrupaciones (2). Todos estos mapas, los generados mediante la DT y el del root se apilan en un mismo mapa de características con P+1 niveles, los cuales se pasan por un *object geometry filter* (Sidebar de la figura) que da como resultado la pirámide de puntuaciones final, de los objetos.

El autor implementó este algoritmo modificando el código ya existente de DPM voc-release5 [7] y usando el framework Caffe [6].

Hay dos formas de entrenar a la red neuronal, en primer lugar se puede tratar como una CNN que se entrena end-to-end con SDG (Stochastic gradient Descent) y backpropagation, y la segunda en dos fases, se desarrolla el DPM como CNN y después se entrena en la

primera fase usando *latent SVM* [1]. El autor se centra en la *latent SVM* ya que dice es la base necesaria para la creación del end-to-end.

Para crear la pirámide, usa como decíamos anteriormente una variante de la SuperVision CNN, y para más tarde comparar los resultados con una R-CNN usa los pesos que fueron entrenados en el *ILSVRC 2012 classification dataset* usando *Caffe*.

Para generar la variante le hace dos modificaciones, la primera es truncar la red quitando la última capa de agrupación, y con ella las capas conectadas (fc_6 ,fc_7 ,fc_8) , de esta manera la salida de la red será la quinta capa convolucional que tiene 256 canales. El segundo cambio, es que antes de cada convolución, con un kernel de tamaño *k* se le añade ceros a la capa de entrada por todos los lados, esto es debido para que todas las entradas y salidas tengan el mismo tamaño espacial. Por lo tanto, y por simplicidad aunque la capa conv5 tenga 256 x 256 pixeles con un paso de 16 pixels, el autor propone que cada imagen se redimensione a 1713 x 1713 pixeles. Esto dice que ayuda a detectar objetos pequeños, aunque dependerá del dataset usado y de cuanto pequeño son estos objetos. Recapitulando, en números, tendrá siete niveles de pirámide y en el primer nivel 108 cells en su lado más grande, y el factor de escala es de $2^{-1/2}$ (raíz de dos), por lo que la pirámide entera tiene 25k cells de salida, por lo que para procesar este algoritmo se necesitará una buena GPU, el autor en concreto usa la NVIDIA Titan Black que tarda 0.5 segundos en computar la conv5.

Por último, y lo que dice el autor es que no es necesario, tener como teníamos en DTDP y HDG un modelo con flip, es decir, parte delantera y parte trasera, y que los resultados de la pirámide de puntuaciones en su mayoría sean negativa, para de esta forma, usar un método de NMS diferente IoU (Intersection over Union) que funciona mejor con conv5. También decir que las partes no están a la doble resolución como en DTDP o HDG, ya que ahora no las busca en la capa de la pirámide que está al doble de resolución. Por lo que si representamos el modelo nos aparecerán a simple vista más pequeñas que en los otros modelos.

Tras la explicación del algoritmo solo queda saber si de verdad se mejora el rendimiento a la hora de detectar personas mediante este algoritmo y el autor compara este nuevo algoritmo con el DTDP obteniendo mejores resultados sobre el dataset VOC 2010. También dice que aunque DP-DPM ofrece buenos resultados, si se ejecutase con R-CNN mejoraría aún más. Por lo que podemos afirmar que cumple con el objetivo que se planteaba, mejorar a los algoritmos de características tradicionales como HOG, aunque a costa de necesitar mayor rendimiento computacional a la hora de procesar las imágenes.

2.5 Conclusiones

Tras estudiar los tres algoritmos ya hemos conseguido tener más claro cómo funciona cada uno de ellos, a parte de la evolución que han tenido, de esta manera podemos llegar a las siguientes conclusiones.

Tanto DTDP como HDG hemos visto que ofrecen buenos resultados, DTDP en entornos en los que las personas no tienen apenas oclusiones con objetos o con las propias personas del entorno, mientras que HDG mejora a este algoritmo en el caso en que se produzcan oclusiones entre las personas o con objetos. Aun obteniendo buenos resultados, el autor de

DP-DPM se propuso mejorar el algoritmo DTDP introduciendo una nueva vertiente en el algoritmo con el afán de seguir la tendencia de uso de redes neuronales, en concreto las CNNs. Y es por ello que como hemos visto, lo consigue, aunque no cuenta con ninguna implementación que trate a esas personas que en algún momento de la secuencia se encuentran ocluidas por otras personas u objetos.

De esta manera, acabando la sección 1 podemos dar paso a como se ha diseñado este algoritmo nuevo, fruto de la motivación antes comentada, y que pretende mejorar los resultados obtenidos por DP-DPM en secuencias de videos donde se produzcan oclusiones.

3 Diseño

3.1 Introducción

En este capítulo procederemos a describir el proceso previo a realizar el desarrollo del nuevo algoritmo, el cual, a partir de ahora lo denominaremos como DP-HDG (DeepPyramid – Hieralchical Detection of person in Groups).

Estudiaremos como desarrollar nuestro algoritmo en base a lo ya existente y decidiremos cuales son las implementaciones necesarias para lograr el objetivo final, que viene siendo mejorar aquellos escenarios en los que nos encontramos con oclusiones con el uso de CNNs.

3.1.1 Lenguaje de programación

El lenguaje de programación será en Matlab y C, debido a que los algoritmos ya implementados están escritos en estos lenguajes, y como reutilizaremos parte del código ya implementado, es la manera más eficiente de desarrollar DP-HDG. La mayoría de código está escrito en Matlab aunque también nos encontramos con scripts escritos en C, los cuales serán necesarios compilar cuando ejecutemos por primera vez el código en Matlab.

Por otro lado como haremos uso de redes neuronales, y es el algoritmo DP-DPM del que partimos, haremos uso del framework Caffe [6] el cual utiliza el autor para desarrollar las CNNs.

Caffe según el autor es la herramienta que proporciona a investigadores y profesionales un framework limpio y modificable para algoritmos de aprendizaje profundo, con una amplia gama de modelos de referencia. Es una biblioteca escrita en C++ y con enlaces a Python y Matlab para desarrollar en estos el despliegue de las redes neuronales convolucionales. Caffe se mantiene por Berkeley Vision and Learning Center (BVLC) con la ayuda de contribuyentes mediante GitHub.

Tanto Caffe como Matlab se instalan sobre un pc proporcionado por el VPU (Video Procesing and Understanding lab) con sistema operativo Linux.

3.1.2 Modelo

Como hemos ido viendo durante la sección Estudio del Arte: el modelo utilizado por los algoritmos ha ido variando a medida que han evolucionado. En el algoritmo DTDP el modelo utilizado es el de la Figura 2-1 compuesto por un root, y por 5 partes deformables que al contrario que DTDP y HDG no están al doble de la resolución.

El autor del HDG cambió el modelo que utilizaba para detectar personas por uno más sofisticado, el cual puede verse en la Figura 2-4, y que proponía el siguiente cambio; el

modelo pasa a tener ocho partes en vez cinco con el propósito de conseguir mejores detecciones. Lo que provoca que hubiera que realizar cambios a la hora de detectar.

Ya en el último algoritmo, el DP-DPM, se utiliza un nuevo modelo que a simple vista puede parecer más extraño que los dos anteriores pero que tiene el propósito de ser mejor a la hora de detectar.

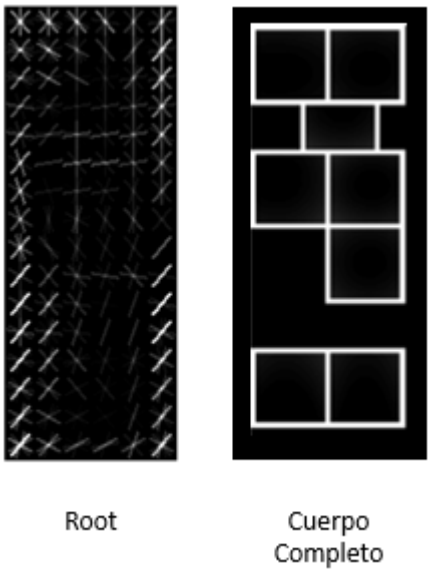

Figura 3-1 Modelo de persona utilizado por DP-DPM y que utilizaremos en DP-HDG.

Como podemos observar, el modelo ha cambiado respecto del HDG al DP-DPM, en primer lugar tenemos el filtro root, y en segundo lugar tenemos solo las cajas correspondientes a las partes.

Vemos que ha cambiado tanto la posición de las cajas de las partes como el tamaño de estas. Si hablamos de la posición ahora tenemos que en la parte donde antes se situaba la cabeza ahora tenemos dos cajas, y donde antes teníamos dos cajas para los hombros solo tenemos una, también podemos ver un ligero desplazamiento de la caja de la pantorrilla. Estas modificaciones se deben a que con el modelo anterior se busca tener una posición de las cajas que se asemejan a una persona vista de frente y con su flip vista de espaldas, y con esta nueva, ya que no hay flip en este modelo, se busca una posición de las cajas que pueda detectar tanto a una persona de lado, de cara y de espaldas.

En cuanto a las cajas de las partes están a la mitad de tamaño que las que presenta el modelo del DTDP o del HDG. Y es debido a que este modelo no tiene las partes del cuerpo al doble de resolución, y es por eso que ahora trataremos los desplazamientos con otros valores.

Todas estas modificaciones se han tenido que tener en cuenta a la hora de desarrollar el algoritmo, que explicaremos más adelante.

3.1.3 Modelo jerárquico de grupos

Con el objetivo de mejorar la detección de esas personas que como venimos hablando sufren oclusiones por objetos o por otras personas, añadimos el primer cambio al DP-DPM, este cambio utilizará la información de la persona principal para mejorar la detección de las secundarias.

Como el modelo de persona ha cambiado, intentaremos replicar lo que el algoritmo HDG hacía para sus desplazamientos con su modelo de persona, pero aplicando los cambios necesarios para que funcione con nuestro modelo.

Si nos fijamos en la Figura 2-5, podemos hacernos una idea de cómo lo replicaremos, lo primero es darnos cuenta que en el nuevo modelo las cajas se encuentran a la mitad de tamaño, por lo que ahora los desplazamientos entre el modelo para buscar alrededor de la persona principal en vez de ser 6 será de 3. En lo demás seguiremos el mismo esquema, mover el modelo hacia la derecha, arriba y la izquierda. Podemos obviar el movimiento de búsqueda por debajo de la MP ya que si una persona se encuentra ocluida, es imposible que sea por la parte de debajo de otra, y como es un caso inexistente se omite.

De esta manera ya sabemos de cara al desarrollo, como implementar el código necesario para esta búsqueda de personas secundarias con el nuevo modelo propuesto.

3.1.4 Modelo jerárquico de partes

Que el modelo haya cambiado, repercute directamente sobre la localización de las partes del cuerpo que usa para detectar, como decíamos en la sección 3.1.2 ahora hay dos cajas en la parte superior y una en la parte de los hombros y la caja de la pantorrilla está desplazada, todo esto con el afán de mejorar a la hora de detectar.

Debido a estos cambios, necesitamos realizar implementaciones en nuestro diseño, ya que como hemos dicho anteriormente, repercute directamente sobre el algoritmo que queremos generar.

En primer lugar tendremos que generar nuevas configuraciones, aunque sigan la lógica que el HDG proponía, al tener diferente localización tendremos que cambiar parte de los vectores que hacen que podamos buscar a una persona con las diferentes configuraciones.

El diseño es parecido, y cómo podemos ver en la Figura 3-2 tendremos cinco configuraciones, la 11 para el cuerpo completo en la que nos quedamos con todas las partes, otra en la que quitamos las cajas correspondientes a los pies que será la configuración 12, la siguiente, la 13, le quitaremos la pantorrilla quedándonos con manos, hombros y cabeza, para la configuración 14 nos quedaremos solo con los hombros y la cabeza, y por último para la configuración 15 nos quedamos con la cabeza, que hemos definido como la caja de más arriba a la izquierda.

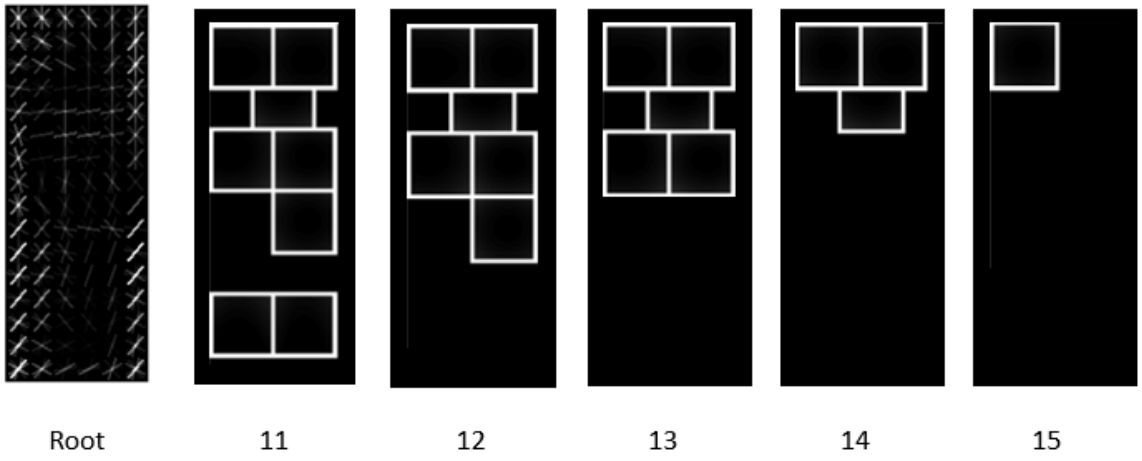

Figura 3-2 Configuraciones elegidas para el nuevo modelo.

Como tendremos que implementar el algoritmo para que busque alrededor de una persona con las configuraciones correspondientes, y como vimos en el HDG, cuando buscas a la SP hay que modificar según busques a la derecha, izquierda o arriba, también modificaremos las partes que buscamos cuando nos encontramos en estos casos, ya que habrá que omitir las partes que creamos que se van a quedar por detrás de la MP y que no se detectarán.

3.1.5 Distribuciones

Si hablamos de distribuciones, también necesitarán cambios, porque tienen relación directa con las partes del cuerpo.

Ya en HDG se tuvieron que calcular las distribuciones de confianza que mostraba cada parte del cuerpo del modelo y el cuerpo entero. En nuestro caso también será así ya que ahora no nos sirve lo que en su día calculó el autor.

Calcular la distribución será necesario debido a que cuando buscamos alrededor de la MP a otras personas, a estas las buscamos con otra configuración diferente y además en otro rango de puntuaciones, por lo que necesitamos crear una serie de distribuciones acumuladas que nos proporcionen esa información que necesitamos.

En estas distribuciones, tendremos entonces la relación entre el score y la probabilidad de encontrar cada configuración, como hay cinco configuraciones y cada configuración, a parte de la MP, tiene tres más para la búsqueda alrededor de esta (derecha, izquierda y arriba) tendremos que generar veinte distribuciones.

Para obtener unos resultados fiables, y que las distribuciones sean homogéneas, se deberán procesar diecinueve videos del dataset creado para el challenge PDbm (People Detection benchmark), los videos están explicados en [10], y que presentan variedad de escenarios, iluminación y complejidad. Estos videos se ejecutan sobre el algoritmo original DP-DPM, de los cuales se sacarán todas las detecciones con un umbral de mínimo -3. Estas detecciones no se obtendrán de manera usual, sino que hará falta sacar el score que cada parte del cuerpo del modelo ha obtenido. Y finalmente generar un archivo de tipo *.mat* con el cual podemos hacer la conversión en el momento en el que sea necesario.

Tras la realización del diseño, en el siguiente capítulo explicaremos de forma detallada como se han conseguido obtener las distribuciones.

3.1.6 Almacenamiento de resultados

Esta parte es cuanto menos importante, y hay que hacer un buen diseño para después poder analizar los resultados obtenidos.

Cada vez que un frame de un video es analizado por nuestro algoritmo DP-HDG, se genera una imagen de scores, que nuestro algoritmo lee y decide donde se encuentran las personas en la imagen original. Por lo tanto, es necesario crear un fichero de salida que guarde todas esas detecciones. Tanto los algoritmos DTDP, HDG y DP-DPM lo hacen de la misma manera, crean un fichero que contiene el nombre de cada frame del video, seguido de cada una de las detecciones del frame, con la posición de esta en la imagen.

Los archivos podrán ser de dos tipos, los hay del tipo .idl (Interface definition lenguage) y de tipo .txt (texto plano), aunque en realidad nos dará igual el formato ya que a la hora de leerlos adaptaremos el código para tratarlos por igual.

Una de las razones por la que también es importante generar estos ficheros, es porque generaremos un script en Matlab que tan solo usando estos ficheros de detecciones y los frames del video, será capaz de mostrar la secuencia visualmente con las detecciones según el umbral que le marquemos.

El proceso de generación de ficheros de salida será explicado de manera más detallada en el siguiente capítulo.

3.2 Conclusiones

Todo algoritmo antes de su desarrollo necesita una fase de diseño, para tener claro cómo queremos que sea y que implementaciones tenga. En este caso el diseño se ha hecho detenidamente, de manera que nos permita conseguir el objetivo de desarrollar un algoritmo que mejore los resultados ofrecidos por DP-DPM en escenarios donde se formen grupos de personas y haya oclusión entre ellas.

Una vez terminado con el diseño, hablaremos de los procesos de desarrollo llevado a cabo para la elaboración del algoritmo, de manera mucho más técnica y explicando a nivel de código las implementaciones requeridas.

4 Desarrollo

4.1 Introducción

En esta sección se muestra todo el desarrollo de implementación del nuevo algoritmo, explicaremos como se ha llegado del diseño anteriormente explicado, hasta un algoritmo capaz de mostrar los resultados que buscamos.

Esta será la parte más técnica del trabajo, así que intentaremos explicar de manera más profunda el modelo, la jerarquía de personas y de partes, y las distribuciones.

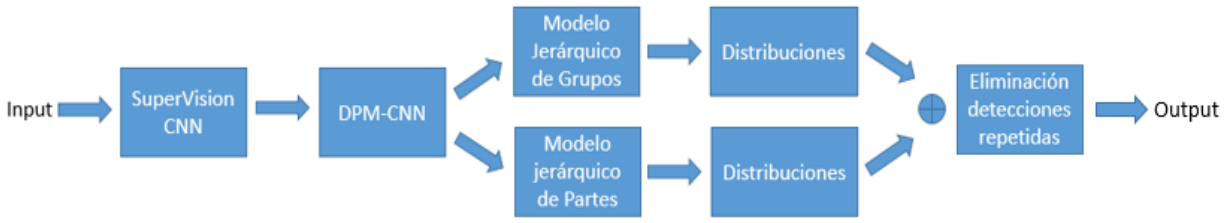

Figura 4-1 Flujo que seguirá una imagen desde que entra hasta su salida.

En la Figura 4-1 podemos ver el proceso del nuevo algoritmo que desarrollaremos, como vemos añadimos la algoritmia referente al modelo jerárquico de grupos, modelo jerárquico de partes, las distribuciones y la eliminación de detecciones repetidas, para llegar al output que serán las detecciones finales.

4.2 Implementación

4.2.1 Instalación de software

En primer lugar, instalamos los programas que fuésemos a utilizar, para ello nos descargamos de GitHub el código del DP-DPM [3] y que proporcionaba manuales para la instalación de Caffe.

Una vez descargado e instalado importamos el proyecto a Matlab y compilamos una primera vez para que aquellos archivos en C puedan ejecutarse desde este programa sin problema. A partir de aquí empezamos a introducir cambios para conseguir el objetivo que buscamos.

4.2.2 Jerarquía de grupos

La primera implementación será una función que modifique el modelo existente que en este caso es el expuesto en la Figura 3-1, a la que llamamos ***modificarmodelo.m*** cuya función principal será la implementación de los vectores del modelo de las diferente configuraciones del cuerpo. A esta función le pasamos como parámetros el modelo, y los desplazamientos que deseamos. Estos desplazamientos son a los que nos referíamos en la sección 3.1.3 cuando hablábamos de las posiciones por las que íbamos a buscar a las personas secundarias alrededor de la persona principal. Como ya explicamos, este modelo no tiene las partes del cuerpo a la doble resolución como lo hacía DTDP o HDG, y que HDG, debido a esto, definía esos desplazamientos de +6 y -6, y es por eso que ahora nosotros dividiremos ese valor entre dos quedándonos esos valores por defecto en +3 y -3. Estos valores son modificables por lo que se pueden modificar si la secuencia lo requiere.

Para guardar estos desplazamientos usamos la variable *anchors_shift* que será un vector de nueve filas por 3 columnas en la que existirá la combinación de todos los posibles valores de estos, en la primera columna el valor del desplazamiento de *x*, en la segunda columna el de *y*, por último en la tercera columna en este caso irá un 0 por defecto ya que ahora no hay doble resolución. También añadimos lo que llamamos step que será el paso utilizado para poder, dentro del mismo rango, detectar más personas. El step será por defecto de 3, pero es un parámetro modificable y se podrá modificar si se requiere.

4.2.3 Jerarquía de partes

Dentro de ***modificarmodelo.m*** también definimos el modelo jerárquico de partes, esto es, definimos las diferentes configuraciones que tenemos para buscar personas y que definíamos en el diseño en la sección 3.1.4.

Definimos un *switch* que según la configuración que hayamos elegido para buscar a la persona principal elige unos vectores específicos. Para cada caso generamos, cuatro vectores que guardamos en la variable model, que es la que tiene todos los valores referentes al modelo, estos vectores son de 1x9 dimensiones y que se rellenan con ‘1’ y ‘0’.

Tendremos un vector para la persona principal, y otros tres para cada lugar donde buscamos a las personas secundarias, a la derecha, a la izquierda y arriba.

El primer valor del vector es el que dice si hay root o no, en nuestro caso siempre hay root por lo que siempre estará a ‘1’, los ocho valores siguientes hacen referencia a las partes del cuerpo que se van a utilizar para buscar a las personas y que tienen las siguientes asignaciones:

1. Root
2. Hombro Izquierdo
3. Hombro Derecho
4. Pie Derecho
5. Mano Derecha
6. Pie Izquierdo
7. Cabeza
8. Mano Izquierda

9. Pantorrilla

De esta manera, si ponemos de ejemplo la configuración 11, en la que se busca a la persona entera, tendremos los siguientes vectores:

model.conf_main = [1 1 1 1 1 1 1 1 1]; → Persona principal
model.conf_groups{1} = [1 0 1 1 1 0 1 0 1]; → Derecha
model.conf_groups{2} = [1 1 0 0 0 1 1 1 1]; → Izquierda
model.conf_groups{3} = [1 1 1 0 0 0 1 0 0]; → Arriba

Este ejemplo sirve para poder entender mejor las partes que omitimos, si buscamos por la derecha, vemos que nos quitamos las partes que están a la izquierda, el hombro izquierdo, la mano izquierda y el pie izquierdo, de esta forma nos evitamos intentar detectar partes que lo más probable es que estén ocluidas. Y de la misma forma haremos tanto si queremos buscar por la izquierda o por arriba, en el caso de buscar por arriba solo nos quedaremos con la cabeza y los hombros.

Todas estas configuraciones, según el modelo o los modelos que elijamos para detectar, se guardan en la variable model, que es la que después usaremos para conseguir los scores de la persona principal y la de las secundarias.

Una vez definido esto, el siguiente paso que sigue el algoritmo es calcular la pirámide de características, es decir le pasamos una imagen de entrada que descomponemos en tres canales, RGB (Red Green Blue), también el modelo a utilizar, en nuestro caso el de la persona que ya hemos definido, y nos devuelve esta pirámide calculada en la CNN. Se ha implementado el algoritmo para que guarde en caché una copia de cada pirámide de características, ya que por frame nosotros hemos tardado alrededor de 50 segundos en calcularlo, y de esta forma lo tendremos disponible al momento.

Una vez se calcula esta pirámide, lo siguiente es calcular las detecciones con la jerarquía de partes y la jerarquía de grupos que hemos definido. El algoritmo pasa por la función gdetect_dp que el autor la define como la que implementa la programación dinámica para calcular las derivadas de alta puntuación usando Object Detection Grammar, y a la cual pasamos por parámetros el modelo y la pirámide.

Lo primero que hace es aplicar la regla de deformación de las partes, después aplica el DT polling (distance transform), y más tarde calcula los puntos de máxima puntuación de las reglas. A continuación, lo que hace es juntar los resultados de cada parte y le suma los offset, el offset es un valor determinado del algoritmo que siempre es igual, y es aquí donde se introduce el primer cambio. Se pasa por ***calculate_main*** que se encarga de detectar de la persona principal todas sus puntuaciones, para al final pasar por ***table_distributions*** que pasa los scores a probabilidad para más tarde poder sumarlo con las personas secundarias.

Una vez están calculados los scores de la persona principal, nos encontramos con la segunda modificación, se pasa a calcular los de las personas secundarias, es decir, aquellas que están ocluidas por la principal. Para ello pasamos por ***calculate_groups,*** es aquí donde entran en juego las configuraciones que antes definíamos de personas, ya que calcula los scores de la persona que busca a la derecha, a la izquierda y arriba, y dependiendo de la configuración buscará unas partes u otras.

4.2.4 Distribuciones

De esta manera ya tenemos las puntuaciones de la persona principal y de las secundarias, pero tenemos un problema, y es que no podemos combinar los mapas de detecciones de la persona principal con la secundaria ya que no contienen el mismo número de partes del cuerpo y por lo tanto de puntuaciones, de esta manera asumimos que una persona que no es visible al 100% sigue siendo persona con la misma confianza que una que se la ve entera, es por ello que el ***calculate_main*** pasaba por la función ***table_distributions.***

Para solucionar este problema nos hemos basado en [9], en primer lugar, hemos elegido diferentes videos, en concreto diecinueve los ya definidos en la sección 3.1.5, con diferentes iluminaciones, complejidad y tamaño de las personas, los hemos pasado por el algoritmo y hemos generado un archivo de salida especial. En este archivo de salida lo que hacemos es poner la posición de la detección con el siguiente formato [x1 y1 x2 y2], lo que quiere decir la posición de la esquina de arriba a la izquierda (x1,y1) y la esquina de abajo a la derecha (x2,y2), y la puntuación que ha dado la detección de la persona, hasta aquí todo normal; pero además añadimos la puntuación del offset, la del root y las puntuaciones de cada parte del modelo, la cabeza, el pie derecho, etc…

Un ejemplo:

(229,286,342,480): 0.7677, -3.5128, 2.6726, 0.3537, 0.2169, 0.0091, 0.3207, 0.0146, 0.2465, 0.2896, 0.1569

Como comprobación de que estas detecciones están bien, si sumamos el offset, con las detecciones de cada parte del cuerpo y el root, el valor esperado es 0.7677, y se cumple siempre.

De esta forma, podemos filtrar por parte del cuerpo y root todas las puntuaciones de todos los diecinueve videos procesados, y sacar la suma acumulada de todas ellas, que después juntaremos en función de las configuraciones que usemos para detectar. En definitiva el objetivo de esto es sacar una función de distribución acumulada, que tenga por eje *x* la probabilidad entre 1 y 0, y en el eje *y* el valor de las detecciones, y poder saber, para cierta configuración cual será el valor real de esa detección cuando por ejemplo no detectamos todas las partes.

Para sacar esta gráfica de relación, hemos generado vectores de x valores por cada parte del cuerpo para después crear la variable X:

X=[S0' model.conf_main{i}.*[S1' S2' S3' S4' S5' S6' S7' S8' S9']];

Con todos esos valores concatenados en un vector con el offset (S0') y las partes según la configuración usada, siendo model.conf_main un vector de 1s y 0s. A este vector es al que queremos hacer la suma acumulada y lo pasamos mediante la función ***cdfcalc()*** de Matlab, que nos da el eje *x* y el *y*.

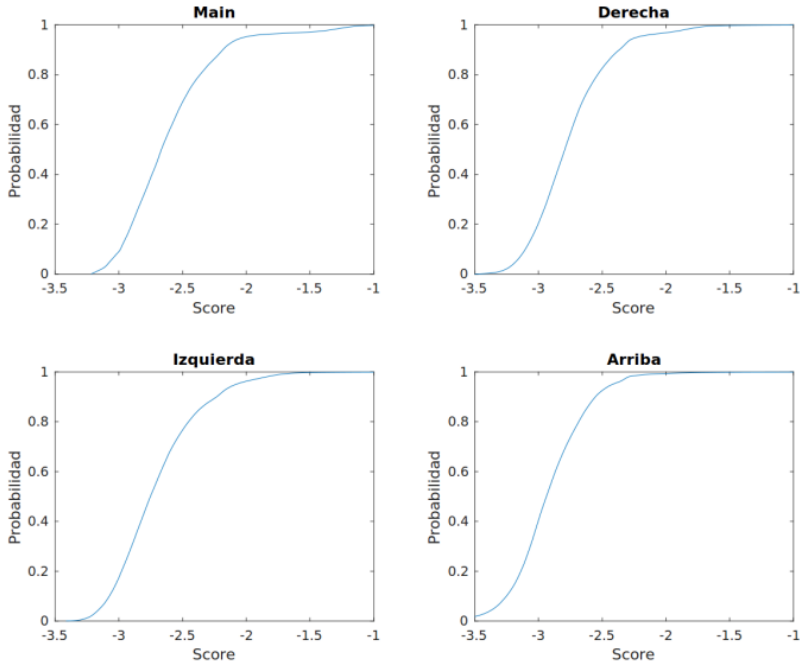

Figura 4-2 Función de distribución de suma acumulada del Cuerpo Entero.

El resultado nos quedaría como en la Figura 4-2, en la que tenemos las funciones de distribución para el Cuerpo Completo, del main, la búsqueda de la derecha, de la izquierda y la de arriba. Se puede observar que son parecidas, pero no iguales, y en realidad es esta diferencia lo que hará que la suma de diferentes modelos pueda llevarse a cabo, ya que la distribución de cuerpo completo tiene un rango ligeramente mayor que las configuraciones sin alguna de las partes del cuerpo en las que esas partes ausentes no suman.

Para simplificar y acelerar el proceso de transformar los scores a probabilidad y viceversa, lo siguiente que hacemos es pasarlos por la función ***fit***, usando el ajuste ***smoothingspline*** que era el que mejor se ajustaba a la curva que queríamos, y que nos devuelve un objeto de tipo ***cfit*** que contiene una curva homogénea sin grandes saltos entre puntos, y que además nos ayuda a ahorrar espacio y a conseguir rapidez en un futuro cuando tengamos que leer los valores de estos objetos.

En total tendremos veinte distribuciones de suma acumulada, cinco configuraciones (11,12,13,14,15) y cada configuración, la persona principal y las de la derecha, izquieda y arriba. Estos valores los guardamos en ***distribuciones.mat*** que será de 5x20, con una fila para el ***cfit*** del eje *x*, otra para el ***cfit*** del eje *y*, otra para la configuración a la que pertenece, la siguiente para el mínimo de la distribución y otra para el máximo de la distribución.

Un ejemplo de la primera columna sería:

1. 1x1 cfit
2. 1x1 cfit
3. [1,1,1,1,1,1,1,1,1,1]
4. -3.2340
5. 2.7351

Una vez que tenemos ***distribuciones.mat*** podemos explicar la función ***table_distributions,*** la función que hemos generado y que será la encargada de cuando le pasemos los scores de la pirámide de detecciones y una configuración, ser capaz de pasar de Score a Probabilidad y de Probabilidad a Score. Esta funcion coge la imagen de detecciones y el ***cfit*** correspondiente, los pasa por la función ***feval*** de Matlab, de esta forma nos convertirá, como decíamos, de Score a Probabilidad y de Probabilidad a Score. Elegimos este método porque en un pincipio lo que hicimos fue indexar las imágenes e ir calculando pixel a pixel esta conversión, lo que disparó los tiempos de ejecución, por eso buscamos la alternativa de usar ***feval*** que es mucho más óptima.

De esta manera, una vez tenemos las puntuaciones de la persona principal con sus partes calculadas en el rango de valores de probabilidad que hemos definido entre 1 y 0, y los scores de las personas secundarias con sus partes del cuerpo también calculadas entre 0 y 1, el siguiente paso será agrupar esas detecciones globales que podemos sumar, ya que están en el mismo rango de valores.

Y por último, necesitamos pasar los valores de scores, los cuales están entre 0 y 1, al rango de scores que irá entre -3 y 1.5. Para ello pasamos por última vez por ***table_distributions*** para hacer el proceso contrario.

Una vez que se han calculado las detecciones y están en el mismo rango, hay que proceder a eliminar detecciones recurrentes. Nuestro algoritmo prueba a detectar personas alrededor de la principal por lo que no es raro que muchas veces detectemos dos veces a la misma persona sin darnos cuenta, por lo que debemos generar una forma de detectar y quitar a esas personas que en realidad están repetidas.
Las detecciones calculadas, se pasan mediante parámetros que tienen la posición relativa de las cajas de las personas y la posición de cada parte del cuerpo detectada, con el mismo formato que antes comentábamos [x1 y1 x2 y2], a parte del nivel de la pirámide donde se ha encontrado esa detección y su puntuación final.

4.2.5 Eliminación de redundancia

Lo siguiente será implementar la función que se encargue de eliminar esas detecciones de las que hablamos, y para ello filtramos de dos formas. Lo primero que hacemos es coger las cajas que se podrían detectar como cabezas de las personas, estas son del modelo presentado en la Figura 3-1 las dos cajas de más arriba. Lo que hacemos es concatenarlas creando una caja de esas dos, quedaría un rectángulo de 6x3, nos quedamos con ese rectángulo y el score de la detección, y para todas las cabezas de todas las detecciones que se han generado le aplicamos un NMS (Non-Maximum Supression) del 0%, lo que quiere decir es que en el caso en que compare dos cajas de las cabezas y detecte que se solapan más de un 0%, se tratará de la misma persona y por lo tanto se quedará con la persona que tenga mayor puntuación de las dos. Si nos fijamos en la Figura 4-3, vemos que hay dos detecciones de cabezas que se solapan, por lo que se quedará con la que mejor puntuación tenga.

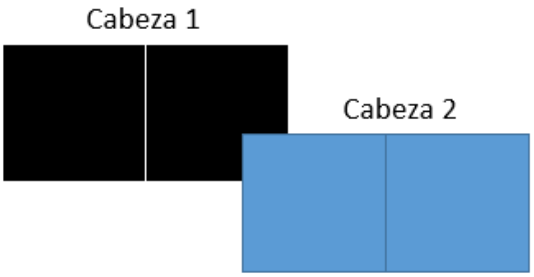

Figura 4-3 Solapamiento entre dos cabezas

El siguiente filtro que aplica esta función es a nivel global de la caja, es decir, ya teniendo filtradas esos solapamientos entre cabezas podría darse el caso que haya dos detecciones en las que la caja del cuerpo entero se solapen entre ellas dando como resultado detecciones repetidas de la mismas persona, en este caso la algoritmia es similar a la de las cabezas. Cogemos las detecciones de las cajas del cuerpo entero restantes de aplicar el NMS a las cabezas, ponemos un valor de NMS del 50% y vemos cuales se solapan más de ese valor, en caso de que superen el 50% de solape, descartaremos la de menor puntuación.

Tal y como podemos ver en la Figura 4-4, las cajas que comparamos se solapan más de un 50%, por lo que al igual que pasaba con la de las cabezas nos quedaremos con la que mejor puntuación tenga de las dos. Decir que para la cabeza siempre se filtra por un valor de NMS 0% ya que hay que ser muy restrictivos, sin embargo, el valor del NMS para las cajas de los cuerpos puede variar, se toma como valor de referencia 50% aunque se consiguen valores parecidos en un rango cercano de NMS.

Con esta implementación conseguimos el objetivo que perseguía, que era eliminar la redundancia de detección de las mismas personas, de tal manera que el algoritmo además gana en la precisión de la posición de las cajas que detecta.

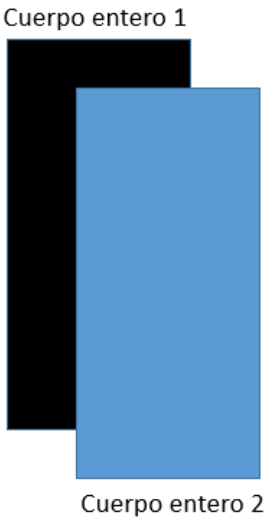

Figura 4-4 Solapamiento entre dos cajas de cuerpo entero

4.3 Conclusiones

Tras terminar de desarrollar todo el código necesario para la implementación del nuevo algoritmo DP-HDG, podemos decir que hasta el momento ha sido la parte más costosa, ya que hemos probado muchas alternativas y sobre todo, el procesamiento de los diecinueve videos en la parte de las distribuciones que ha llevado bastante tiempo.

Ahora el siguiente paso será el de la integración, pruebas y resultados, el paso definitivo en el que sabremos si todo lo hecho hasta el momento de verdad sirve para detectar mejor a las personas en secuencias de videos con grupos de personas.

5 Integración, pruebas y resultados

5.1 Introducción

Una vez tenemos el desarrollo completado, pasamos a la parte más analítica del trabajo, y es la de sacar resultados. En esta sección sabremos si el trabajo realizado cumple los objetivos marcados al principio.

Analizaremos diferentes videos para probar las diferentes configuraciones que se pueden utilizar sobre nuestro algoritmo, y veremos cuáles son los parámetros óptimos para detectar a esas personas que se encuentran ocluidas.

5.2 Elección de dataset

La elección del dataset a utilizar debe ser a conciencia, debido a que necesitaremos uno que ya haya sido puesto a prueba anteriormente y que además sea variado.

Para nuestro caso utilizaremos el dataset PETS2009. Las razones son simples, en primer lugar este dataset ha sido utilizado por los algoritmos DTDP y HDG, por lo que ya tenemos unos valores de referencia con los que comparar el algoritmo DP-DPM y nuestro algoritmo DP-HDG. También, otra de las razones por las que lo elegimos es porque ofrece dentro del mismo dataset diferentes factores meteorológicos que modifican la iluminación, y por, también hay secuencias con más complejidad que otras, además de tener un gran número de personas que se oclusionan constantemente, lo cual nos servirá para sacar conclusiones acerca de en qué escenario nuestro algoritmo pueda obtener mejores resultados y en cuales peores. Cabe destacar que aparte del dataset PETS2009, haremos uso de nuestro algoritmo en los videos tud-campus-sequence y tud-crosssing.

Todos estos videos cuentan además con un archivo llamado Grand-Truth, el cual tiene todas las detecciones reales que aparecen en cada secuencia de video, con las que luego compararemos nuestro algoritmo para conseguir su rendimiento real.

5.3 Evaluación

Para evaluar los videos, seguiremos un procedimiento bien definido, que dividiremos en tres partes. La primera consistirá en HPG (Hierarchy of persons in groups) ,la segunda en HBP (Hierarchy of body parts) y la última en HDG (Hierarchical detector in groups).

Todos los videos se evaluarán mediante la curva ROC (Receiving Operating Characteristic) que la usaremos a modo de diagnóstico de rendimiento de nuestro algoritmo, y de los que partimos. Esta curva hace la relación entre precisión y recall, siendo la precisión el un valor entre 0 y 1 que nos dice cómo se ajustan las cajas de las personas que detectamos a estas, y siendo el recall el número de personas que detectamos.

El área bajo la curva serán los valores de detección con los que evaluaremos.

5.3.1 Hierarchy of persons in groups (HPG)

En primer lugar, lo que queremos ver es cómo se comporta al algoritmo a las diferentes configuraciones de la jerarquía de grupos. Primero viendo los resultados de forma exhaustiva en la secuencia S2L1 y después con todo el dataset. Usaremos las diferentes configuraciones de los ***anchor_shift,*** siempre con una nueva configuración que será la 16, y que no tiene en cuenta la jerarquía de partes, por lo que los cuatro vectores main, derecha, izquierda y arriba, serán todo 1s.

Definiremos los siguientes parámetros, en primer lugar los desplazamientos en el eje *x* y el eje *y* (|Δx|,|Δy|), *l* que será el número de personas que busca alrededor de la principal y varía entre 9 y 81, y *s* que es el step utilizado que usa para buscar entre personas secundarias.

Anchor-shift configuration	**A**	**B**	**C**	**D**	**E**	**F**
|Δx|	3	3	6	6	6	6
|Δy|	3	3	3	3	6	6
s	2	3	2	3	2	3
l	25	9	45	15	81	25
HPG	0.6400	0.6484	**0.6495**	0.6482	0.6456	0.6475

Tabla 1: Resultados HPG para la secuencia S2L1

En la Tabla 1 vemos para la secuencia PETS2009-S1L1-2 los resultados obtenidos con las diferentes configuraciones del ***anchors-shift***, si nos fijamos, no varía mucho según las diferentes configuraciones.

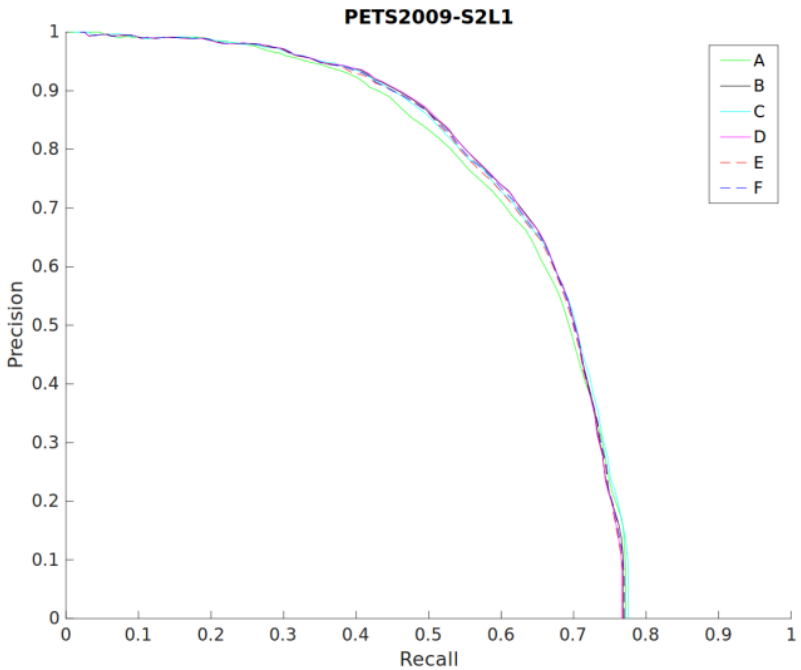

Figura 5-1 Curva ROC de la secuencia S2L1 con HPG

Por último podemos comparar todos los resultados con el algoritmo original DP-DPM, en la Tabla 2 y observamos como en todos los videos mejoramos los resultados detección, respecto al algoritmo original. Esto se debe a que el modelo jerárquico de grupos, busca bien alrededor de la persona principal y encuentra a personas secundarias que se encontraban ocluidas y que el algoritmo original no era capaz de detectar.

		DP-DPM	**DP-HPG**					
Secuencia	**Complejidad**		A	B	C	D	E	F
S1L1-1	**Medium**	0.4104	0.4420	0.4420	**0.4439**	0.4429	0.4413	0.4412
S1L1-2	**Medium**	0.4202	**0.4684**	0.4631	0.4657	0.4641	0.4649	0.4660
S1L2-1	**High**	0.2838	0.3008	0.3120	0.3054	0.3126	0.3132	**0.3139**
S1L2-2	**High**	0.2895	0.3187	0.3225	0.3262	0.3255	0.3268	**0.3276**
S2L1	**Low**	0.5639	0.6400	0.6484	**0.6495**	0.6482	0.6456	0.6475
S2L2	**Medium**	0.3392	0.3689	0.3771	0.3742	0.3772	**0.3817**	0.3815
S2L3	**High**	0.2494	0.3069	0.3102	0.3094	**0.3154**	0.3111	0.3121
S3MF1	**Low**	0.7598	0.8140	0.8226	0.8294	0.8270	**0.8353**	0.8279
Campus	**Low**	0.6504	**0.6670**	0.6621	0.6652	0.6616	0.6653	0.6613
Crossing	**Low**	0.6909	0.6901	**0.7058**	0.6982	0.7044	0.7009	0.7034

Tabla 2 Resultados con las diferentes configuraciones para los diferentes videos.

5.3.2 Hierarchy of body parts (HBP)

Una vez vistos los resultados sobre la jerarquía de grupos, lo siguiente es probar cómo se comporta la jerarquía de partes del cuerpo.

Para ello nos quedaremos con la configuración que mejor resultado nos ha dado con HPG, y probaremos los diferentes casos de configuraciones que definimos en la sección 3.1.4, y además también probaremos a combinar las detecciones de diferentes configuraciones sobre la misma secuencia.

	DP-DPM	DP-HBP									
Secuencia		11	12	13	14	15	11-15	12-14	12,13	12,14	13,14
S1L1-2	0.410	0.438	**0.455**	0.450	0.420	0.366	0.437	0.445	0.454	0.445	0.434
S1L1-2	0.420	**0.467**	0.463	0.456	0.426	0.366	0.457	0.451	0.458	0.452	0.445
S1L2-1	0.284	**0.304**	0.287	0.280	0.260	0.203	0.280	0.276	0.285	0.275	0.272
S1L2-2	0.289	0.321	**0.337**	0.331	0.309	0.274	0.319	0.334	0.337	0.333	0.327
S2L1	0.564	0.650	**0.664**	0.645	0.580	0.455	0.634	0.634	0.656	0.632	0.619
S2L2	0.339	0.380	0.403	**0.403**	0.386	0.352	0.379	0.396	0.401	0.396	0.397
S2L3	0.249	**0.310**	0.304	0.299	0.268	0.218	0.305	0.295	0.302	0.293	0.292
S3MF1	0.760	**0.828**	0.819	0.776	0.737	0.596	0.816	0.795	0.810	0.793	0.764
Campus	0.650	0.665	**0.693**	0.689	0.685	0.673	0.669	0.690	0.692	0.690	0.687
Crossing	0.691	0.705	0.734	**0.737**	0.734	0.726	0.705	0.733	0.734	0.734	0.736

Tabla 3 Resultados HBP para las diferentes secuencias

Si nos fijamos en los valores, vemos como en todos los casos mejoramos la detección del original, y también de los valores calculados en HPG.

En negrita están los valores más altos obtenidos, y siempre están entre la configuración 11, 12 o 13. Viendo los resultados podemos decir que usando la jerarquía de partes del cuerpo también mejoramos las detecciones. Ya que aparte de la jerarquía de grupos ahora al añadir la jerarquía de partes conseguimos aumentar la puntuación de las partes que no están ocluidas en las personas secundarias.

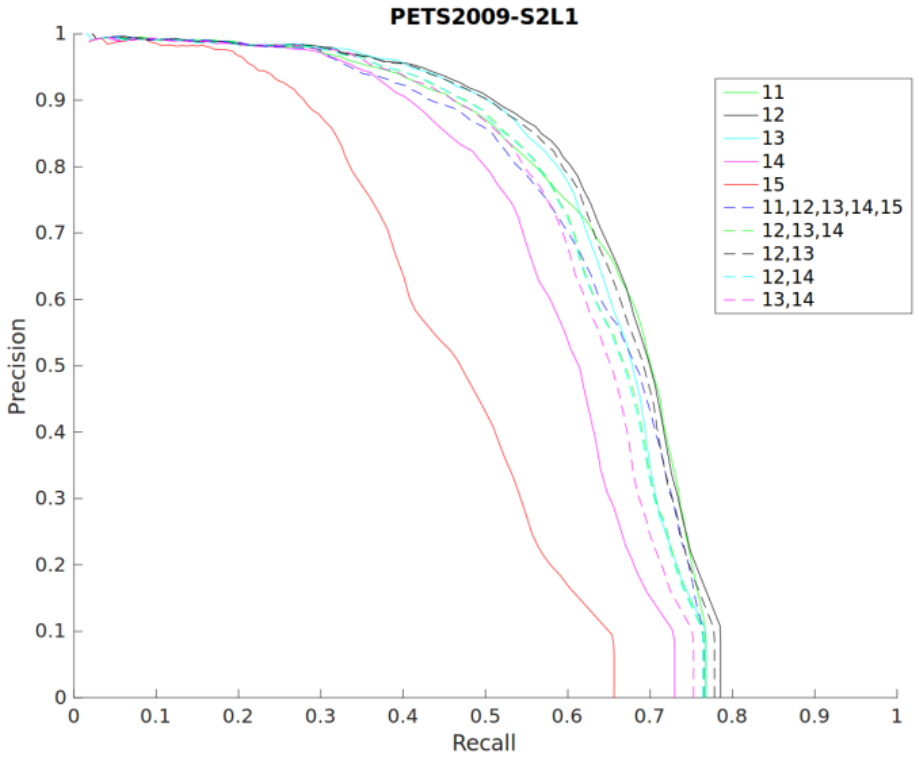

Figura 5-2 Curva ROC para la secuencia S2L1 con HBP

La Figura 5-2 es un buen ejemplo, ya que vemos como detectamos muy bien para las configuraciones 11, 12 y 13, sin embargo baja mucho con las de solo cabeza, y cabeza y hombros, y además vemos como baja la precisión cuando juntamos varias configuraciones, ya que habrá un mayor número de falsos positivos.

5.3.3 Hierarchical detection in groups (HDG)

Una vez ya hemos obtenido los resultados para HPG y HBP, lo siguiente es probar los mejores resultados obtenidos en estas dos anteriores y probar diferentes valores que obtendremos aplicando un NMS más permisivo o menos.

Los valores de NMS solo variarán para el solape entre cajas del cuerpo entero, entre 50% hasta 99%, en saltos del 5%. Para el NMS de las cajas de las cabezas siempre lo mantendremos en 0%, no permitiendo solape entre estas.

	DP-DPM	DP-HDG										
Secuencia		**0,5**	**0,55**	**0,6**	**0,65**	**0,7**	**0,75**	**0,8**	**0,85**	**0,9**	**0,95**	**0,99**
S1L1-1	0.410	0.455	0.464	0.469	0.474	**0.480**	0.477	0.464	0.454	0.445	0.408	0.407
S1L1-2	0.420	0.467	0.474	0.479	0.487	**0.491**	0.483	0.478	0.470	0.465	0.449	0.449
S1L2-1	0.284	0.304	0.310	0.315	0.321	**0.332**	0.330	0.325	0.318	0.311	0.295	0.296
S1L2-2	0.289	0.337	0.348	0.358	0.368	0.373	**0.374**	0.363	0.359	0.354	0.328	0.324
S2L1	0.564	0.664	0.665	**0.667**	0.666	0.665	0.660	0.651	0.643	0.636	0.616	0.615
S2L2	0.339	0.403	0.408	0.415	0.419	**0.426**	0.425	0.415	0.409	0.405	0.381	0.380
S2L3	0.249	0.310	0.316	0.320	0.322	**0.325**	0.325	0.314	0.307	0.299	0.276	0.275
S3MF1	0.760	0.828	0.830	0.835	**0.837**	0.828	0.820	0.806	0.796	0.782	0.754	0.754
Campus	0.650	0.693	0.696	0.693	0.693	0.701	0.695	0.699	0.704	**0.707**	0.690	0.701
Crossing	0.691	0.737	0.734	0.736	0.737	0.739	**0.740**	0.736	0.734	0.732	0.722	0.722

Tabla 4 Resultados de HDG para las diferentes secuencias

Los resultados obtenidos son buenos, ya que mejoramos en todas las secuencias, en los resultados obtenidos en HPG y HBP, podemos ver como en la mayoría el rango de mejores resultados está entre 0,6 y 0,75.

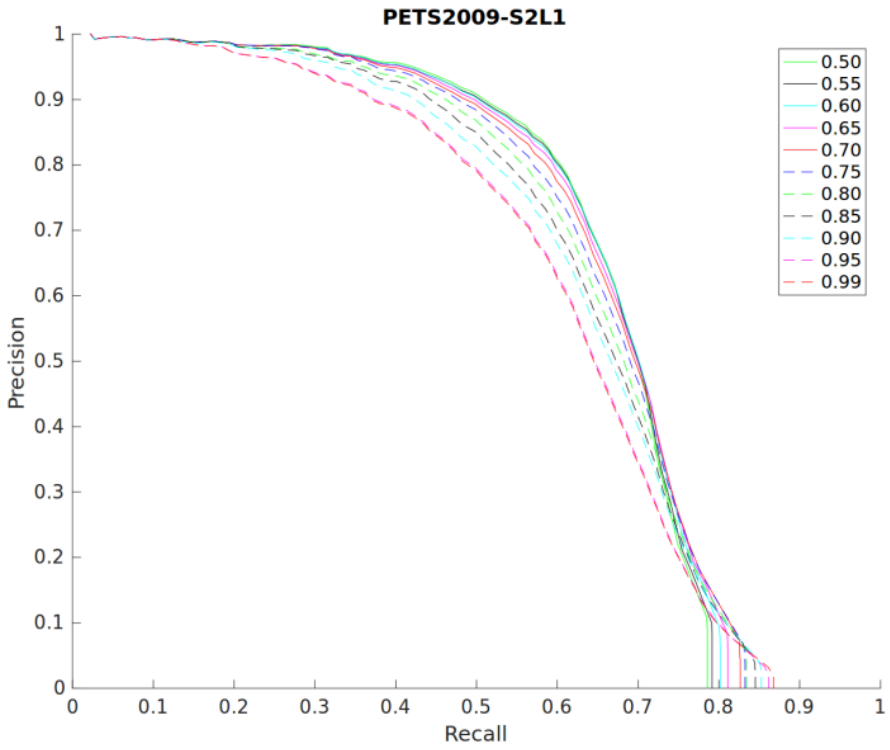

Figura 5-3 Curva ROC para la secuencia S2L1 con HDG

Observando la gráfica nos damos cuenta cómo según aumentamos la permisividad del NMS baja la precisión en la que detectamos a las personas, sin embargo, mejoramos en el recall debido a que tendremos más detecciones, y que puede ser que aunque no sean precisas, detecten a personas.

5.4 Comparación de algoritmos

Por último queda hacer una comparación entre los algoritmos de los que hemos hablado en la sección Estudio del Arte:

Para ello generamos una tabla comparando los resultados obtenidos por DTDP, HDG, DP-DPM y DP-HDG, para las diferentes secuencias.

Secuencia	DTDP	HDG	DP-DPM	DP-HDG
S1L1-1	0.628	0.726	0.410	0.480
S1L1-2	0.734	0.848	0.420	0.491
S1L2-1	0.479	0.679	0.284	0.332
S1L2-2	0.494	0.653	0.289	0.374
S2L1	0.934	0.949	0.564	0.667
S2L2	0.664	0.809	0.339	0.426
S2L3	0.558	0.738	0.249	0.325
S3MF1	0.930	0.942	0.760	0.837
Campus	0.765	0.791	0.650	0.707
Crossing	0.854	0.855	0.691	0.739

Tabla 5 Resultados de todos los algoritmos

Si comparamos los resultados, vemos como en todas las secuencias de video nuestro algoritmo DP-HDG mejora al algoritmo del que partíamos. Sin embargo, este algoritmo no mejora los resultados del DTDP ni del HDG.

5.5 Conclusiones

Tras ver los resultados obtenidos podemos sacar la siguiente conclusion, y es que el objetivo principal de este TFG se ha cumplido, ya que mejoramos en todas las secuencias el valor obtenido tanto en precisión de las cajas de detección como el total de personas que detecta.

Por otro lado, también nos hemos dado cuenta que el algoritmo DP-DPM no hace competencia ni siquiera al DTDP que obtiene unos valores más bajos que el HDG, e incluso en ningún caso nuestro algoritmo DP-HDG lo mejora. La razón que hemos visto es que DP-DPM no consigue buenos resultados si los objetos a detectar son muy pequeños en la imagen, y se comporta mejor cuando estos ocupan mayor parte de esta. Este efecto lo hemos podido comprobar al obtener las detecciones sobre el dataset PETS2009 en los que las personas eran demasiado pequeñas. Para los videos Campus y Crossing, las personas tenían tamaños más grandes y aun así se ha conseguido mejorar las detecciones considerablemente.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

Empezábamos este Trabajo de Fin de Grado hablando de la creciente popularización de las Redes Neuronales en diferentes ámbitos de las tecnologías. Precisamente, parece ser una moda querer hacer todo mediante estos algoritmos de Deep Learning.

Si hablamos del tratamiento de imágenes, el tipo de Redes Neuronales más utilizadas son las CNNs, de las cuales hemos hablado a lo largo de las diferentes secciones, y lo que ha querido hacer el autor del algoritmo DP-DPM ha sido eso, usar las CNNs para intentar mejorar las detecciones de todo tipo de objetos.

Viendo los resultados obtenidos, llegamos a la conclusión de que aún hay mucho margen de mejora para la implantación de detección de personas con CNNs, hemos obtenido unos resultados más bajos en el algoritmo DP-DPM que en el DTDP que usaba características tradicionales como HOG.

Sin embargo, cabe destacar que el objetivo principal de este TFG se ha cumplido, y con buenos resultados. Hemos conseguido mejorar los resultados obtenidos con DP-DPM en nuestro algoritmo DP-HDG. Todas las secuencias de video analizadas han dado valores más altos, y hemos probado que la jerarquización de grupos de personas y la jerarquización de partes del cuerpo a la hora de detectar personas que se encuentran en oclusión es eficiente.

Si analizamos el costo de mejorar este tipo de algoritmo frente a la ganancia de detecciones que obtenemos, mediante las técnicas que hemos usado a lo largo de este TFG, se puede decir que siempre será una alternativa a tener en cuenta.

6.2 Trabajo futuro

Las aplicaciones que se nos ocurren de trabajo futuro son incontables, aunque para la mayoría supondrían un gran costo llevarlas a cabo. Por lo que proponemos, desde nuestro punto de vista, las opciones más óptimas y realistas.

En primer lugar, proponemos estudiar la posibilidad de intentar aumentar el número de personas del grupo, en nuestro caso son dos, pero sería interesante estudiar el caso en el que los grupos de personas sean mayores y hacer un algoritmo que sea capaz de detectar, por ejemplo, a la persona que pueda estar ocluida por la persona secundaria.

Otra aplicación interesante que se podría llevar a cabo es estudiar otros modelos de personas con diferentes ubicaciones de las partes del cuerpo y ver si mejora la detección de personas en oclusión.

También creemos que se podría estudiar la forma de mejorar que este algoritmo detecte mejor a objetos más pequeños, que hemos visto que es una de sus debilidades.

Con respecto al coste computacional, también podría ser de utilidad buscar formas de optimización del algoritmo, ya que hoy en día es necesario una GPU si se quiere tratar imágenes con CNNs con cierta rapidez.

Por último, este algoritmo podría destinarse al uso en la vida real, es decir, implantarlo como software y utilizarlo como detector de personas en situaciones del día a día. Estas situaciones pueden ser, por ejemplo, contar el número de personas que transitan el centro de una ciudad en un día. Otro uso posible puede ser la video vigilancia, usarlo para evitar robos e identificar el número de personas intrusas.

Referencias

[1] P. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(9):1627–1645, 2010.

[2] García-Martín, Á., Sánchez-Matilla, R. & Martínez, J.M. SIViP (2017) 11: 1181. https://doi.org/10.1007/s11760-017-1073-z

[3] R. Girshick, F. Iandola, T. Darrell, J. Malik. Deformable Part Models are Convolutional Neural Networks. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 437-446

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition, volume 1, pages 886–893, 2005.

[5] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In NIPS, 2012.

[6] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In Proc. of the ACM International Conf. on Multimedia, 2014.

[7] R. Girshick, P. Felzenszwalb, and D. McAllester. Discriminatively trained deformable part models, release 5. http: //www.cs.berkeley.edu/˜rbg/latent-v5/.

[8] García-Martín, Á., Sánchez-Matilla, R. & Martínez. Proyecto Final de Carrera. Detección jerarquica de grupos de personas.

[9] A.Garcia-Martin, R.Heras,and T.Sikora. A multi-configuration part-based person detector. 11th of International Conference on Signal Processing and Multimedia Applications, 2014.

[10] A. García-Martín, J.M. Martínez, J. Bescós, "PDbm: People detection benchmark repository", Electronic Letters Volume 51, Issue 7, p. 559 – 560 DOI: 10.1049/el.2014.3795 , Print ISSN 0013-5194, Online ISSN 1350-911X

Glosario

CNN	Convolucional Neuronal Network
DTDP	Discriminatively Trained Deformable Part-based model
HDG	Hierarchical Detection of persons in Groups
DP-DPM	Deep Pyramid – Deformable Parts Model
DP-HDG	Deep Pyramid – Hierarchical Detection of persons in Groups
SVM	Support Vector Machine
HOG	Histogram of Oriented Gradient
NMS	Non-Maximum Supression
ROC	Receiving Operating Characteristic